UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/698,956	10/31/2003	Martin Scholz	16104-014001 / 2003P00684	8804

32864 7590 09/28/2010
FISH & RICHARDSON, P.C.
PO BOX 1022
MINNEAPOLIS, MN 55440-1022

EXAMINER	
DUNN, DARRIN D	

ART UNIT	PAPER NUMBER
2121	

NOTIFICATION DATE	DELIVERY MODE
09/28/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PATDOCTC@fr.com

Office Action Summary	Application No.	Applicant(s)	
	10/698,956	SCHOLZ ET AL.	
	Examiner	Art Unit	
	DARRIN DUNN	2121	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 July 2010*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. The Office Action is responsive to the communication filed on 07/16/2010.

2. Claims 1-20 are pending in the application.

Response to Amendment

3. The amendment, filed 7/16/2010, had been entered.

Response to Arguments

4. Applicant's arguments with respect to claims 1-20 have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. The factual inquiries set forth in *Graham* v. *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.

4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

5. Claims 1-8, 10-12, and 14-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Coker et al. (USPN 2007/0250840) in view over Lee et al. (USPN 20040088700) in view over Balducci et al. (USPN 20040103174) and in further view over Kinoshita et al. (PGPub 20030055792)

6. As per claims 1 and 10, Coker et al. teaches which code (i) is configured to be stored on the client device and be executed during each of subsequent communications between the client device and the server device ([0307-0309] e.g., client includes a component called a busy state manager configured to monitor and inform a user of a status and progress of the submitted request), and (ii) when executed blocks the client device from receiving user input during the communications between the client device and the server device ([0309] e.g., client can inform the user that request processing has started and lock the user interface), determines whether any of the communications between the client device and the server device lasts longer than a specific time, and, upon determining that the specific time has been exceeded, causes a message provided in the code to be presented to a user of the client device ([0309-0310] e.g., upon determining that the request from the client may take a long time to process, the server will notify the client accordingly....the client can update the progress bar to show how much of the task has been completed at that point in time.

However, Coker et al. does not teach the server providing executable code to the client computer, i.e., providing the busy-state manager component to a client computer). Lee et al. teaches a system for automatically installing software on a client via a server ([ABSTRACT], [FIG 1])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to install client components using software stored on a server. Lee et al. teaches that enterprises employ client-server models to facilitate the configuration of a client computer via downloading necessary application software ([0004-0006]). Coker et al. teaches that various types of clients can be supported....the various types of clients including remote clients ([0063]), and in addition, teaches that clients can download a subset of server's data to use locally ([0070]). Therefore, in client-server interactions, as taught by Lee et al. it would have been obvious to enable a server to provide necessary components to remote clients to facilitate server-client interactions, including downloading necessary components to a client. Here, it would have been obvious to download a busy-state manager component to a remote client via an application server.

However, Coker et al., as modified, does not teach that the client device is configured to engage in communications with the server device for a plurality of application programs. Coker et al. does teach that the client computer makes requests to the server such that particular tasks may be executed on the server, and in response to the client request the client computer is locked ([0309-0310] e.g., it is interpreted that the client request is not tied to any particular program such that any program on the client computer that should make a request to the server for server processing would result in the client computer becoming locked). Balducci et al. teaches an application program, i.e., Microsoft Outlook, on the client computer ([0024]) and in addition teaches that the client may delete content within the server ([0075]).

Therefore, it would have been obvious to one of ordinary skill in the art to apply the "lock mechanism" (e.g., client code used to lock the client computer) , as taught by Coker et al., to be

applicable to any program (e.g., Outlook) on the client computer that would require external server processing. Since a client may send a delete request, for example, to a server that could take longer than a reasonable time (e.g., server is backed up), it would have been obvious to apply the "lock mechanism" to any program making such a request that would require the client to be locked. During the period in which the client is locked, the user would be informed that a request is taking longer than expected, as taught by Coker et al. Balducci et al. illustrates that a an application program, such as Outlook, may send a request to the server, in turn locking the client during this task, as taught by Coker et al. (e.g., as in the case of emptying a folder on the server or synchronizing a client and server)

However, Coker et al. does not teach determining whether at least a threshold period of time has already elapsed while waiting for each of the communications between the client device and the server device to finish and upon determining that the threshold period of time has already elapsed, causes a message provided in the code to be presented to a user of the client device. Coker et al. does teach a) providing a message using the code to be presented to a user of the client device [0309-0310] e.g., the code is configured to provide a status indicator) but is silent as to a message displayed in response to the above limitations. Kinoshita et al. teaches the pertinent problem of measuring a time lapse in response to a transmission of a request after a predetermined time, i.e., threshold period, then displaying a message notifying a user indicating a process is aborted ([0187]) upon failing to receive a result after the predetermined time. Coker et al. teaches sending a request a server, locking the client interface, and informing the client the request may take a long time ([0309])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to set a predetermined amount of time for the request to be processed by a server, and upon the request failing to be processed in the predetermined time, notifying the user of the state of the request. Coker et al. teaches that a user may wait for an undisclosed period of time while a request is processing in addition to having the user interface locked. Since a predetermined time is indicative of a how long a process should take to complete, it is obvious to notify a user that the process will not complete in the predetermined so as to apprise the user of the status of their request.

7. As per claim 2, Coker et al. teaches the method of claim 1, wherein the executable code is client-side framework code provided from the framework code in the server device that controls communication between the server device and the client device ([0306-0310] e.g., busy state manager component is stored on the client, i.e. client side framework, provided from the framework code in the server device (e.g., as modified, an application server provides the busy state component to the client), where the framework code controls communication between the client and server, i.e., client is informed by the server communication will last longer than expected, and in response, the client user interface is locked. Controlling communication is interpreted as corresponding to informing the client device of a process status)

8. As per claim 3, Lee et al. teaches the method of claim 1, further comprising providing the executable code in response to the server device receiving a request from the client device to launch at least one of the application program capable of initiating the communications ([0011] e.g., automatically installing software on a client via a client login request. As interpreted, a login request is an application program that initiates the request for the executable code. As

applied to Coker et al., the busy-state component could be downloaded in response to the client request for the component by following the steps provided in paragraph 0011). Once the "lock mechanism" is installed in the client computer (e.g.., code may be installed on the client computer via a provisioning service), any program that could make a request to a server that requires a "lock" to be implemented during the request could be launched. For example, a user makes a delete request to the server using Outlook. The client is locked during the delete request. The client code would lock the client during the use Outlook delete request to the server. If the request takes longer than expected, the client informs the user.

9. As per claim 4, Coker et al., as modified, teaches the method of claim 3, further comprising providing application program code from at least on the application programs to the client device wherein the message is an over-definition of a default message that would otherwise be presented. (As modified by Coker et al., [0309], Outlook provides a status bar, i.e., code, during a delete request. The status bar is displayed to the user with a progress indicator, i.e., over-definition. As per the applicant's background section, "sudden displays of messages and sudden disappearances" are unnecessary. As applied to Coker et al., it would have been obvious to display the indicator when the request is going to take longer than expected and to not display the progress bar as to avoid unnecessary disruptions, i.e., "otherwise be presented." Furthermore, as modified by Kinoshita et al., a message indicating that a process if aborted is alos interpreted as an over-definition of a default message. Over-definition of a default message is interpreted as a message comprising a unique notification.

10. As per claim 5, Coker et al. teaches the method of claim 1, wherein the threshold period of time has already elapsed, supra claim 1, but does not teach the threshold period of time lapse

is due network delays, server-side-delays. Coker et al. does teach wherein a communication lasts longer than the specific time due to network delays, server-side delays, or combinations thereof ([0308], [0309] e.g., delays due to request waiting in the queue, i.e., server-side delays).

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to determine that a predetermined time, i.e., time period from start of request to receiving processing results, has lapsed to server-side delays. It is foreseeable, as per Coker et al., that a server will not process a request on time due to pending requests. In effect, due to requests taking a long time to complete, it is possible that the predetermined time may elapse for a client processing request. The motivation is to take in account that unnecessary delays on a server side will contribute to failing to process a request within a predetermined time period, resulting in user inconvenience. In effect, by informing a user that a request will not be processed due to the expiration of a predetermined time, the user may find alternative resources from which to process their request.

11. As per claim 6, Kinoshita et al teaches the method of claim 1, wherein the threshold period of time elapses when the client device has not displayed a server response within the trheshold period of time, as measured from a first time when the client device requests information from the server to a current time during which the client device is waiting for the response from the server device ([0187])

12. As per claim 7, Coker et al. teaches the method of claim 1, wherein the executable code ceases to block the client device from receiving user input after each communication has ended ([0310] e.g., client continues to lock the interface until the request processing is completed)

13. As per claim 8, Coker et al. teaches the method of claim 1, wherein the executable code causes the message to be presented on the client device during one of the communication and causes the client device to cease presenting the message after that communication has ended ([0310] e.g., as best understood, the busy manager causes the notification to be presented to the user and causes the client to cease presenting the message when the request processing is completed)

14. Claims 9, 13, and 19-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Coker et al. (USPN 2007/0250840) in view over Lee et al. (USPN 20040088700) and in view over Balducci et al. (USPN 20040103174) in view over Kinoshita et al. (USPN 20030055792)in view over Logston et al. (USPN 6687735) and in further view over Conrad (USPN 20020105914)

15. As per claim 9, Coker et al., as modified, teaches the method of claim 1, using a predetermined, i.e., threshold period of time, but does not teach setting a threshold period of time based on at least one selected from the group consisting of: a roundtrip time for a communication between the server device and the client device, typical roundtrip times for communication between the server device and the client device, a roundtrip time expected by at least one user of the client device, and combinations thereof. Logston et al. teaches setting a "sent time" parameter for estimating system latency where latency is measured as round-trip latency ([Col 25 lines 55-67]). Conrad teaches setting a time value using the measured latency ([ABSTRACT])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to set a predetermined time, as per Kinoshita et al., to be based on the expected

round-trip latency to account for delay time associated with sending and receiving information. Since a total request time includes the time the request takes to arrive to the server, the actual processing time, and time to be returned to the client, it is obvious to set a predetermined time to account for latencies so as ensure the predetermined time is not understated.

16. As per claim 11, Coker et al., as modified, teaches a method of informing a user about communications between a client device and a server device, the method comprising:

storing the executable code on the client device, the executable code configured to be executed during each of subsequent communications between the client device and the server device ([0306-0310] e.g., busy-manager component);

blocking, per the executable code, the client device from receiving user input during its communications with a server device ([0309-0310]);

determining whether any of the communications lasts longer than a specific time; and presenting, per the executable code, a message provided in the code to a user of the client device upon determining that any of the communications lasts longer than the specific time ([0310] e.g. server notifies client that the request may take a long time to finish. Please see discussion below for newly amended limitations)

However, Coker et al. does not teach the server providing executable code to the client computer, i.e., providing the busy-state manager component to a client computer). Lee et al. teaches a system for automatically installing software on a client via a server ([ABSTRACT], [FIG 1])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to install client components using software stored on a server. Lee et al. teaches that

enterprises employ client-server models to facilitate the configuration of a client computer via downloading necessary application software ([0004-0006]). Coker et al. teaches that various types of clients can be supported....the various types of clients including remote clients ([0063]), and in addition, teaches that clients can download a subset of server's data to use locally ([0070]). Therefore, in client-server interactions, as taught by Lee et al. it would have been obvious to enable a server to provide necessary components to remote clients to facilitate server-client interactions, including downloading necessary components to a client. Here, it would have been obvious to download a busy-state manager component to a remote client via an application server.

However, Coker et al., as modified, does not teach that the client device is configured to engage in communications with the server device for a plurality of application programs. Coker et al. does teach that the client computer makes requests to the server such that particular tasks may be executed on the server, and in response to the client request the client computer is locked ([0309-0310] e.g., it is interpreted that the client request is not tied to any particular program such that any program on the client computer that should make a request to the server for server processing would result in the client computer becoming locked). Balducci et al. teaches an application program, i.e., Microslft Outlook, on the client computer ([0024]) and in addition teaches that the client may delete content within the server ([0075]).

Therefore, it would have been obvious to one of ordinary skill in the art to apply the "lock mechanism" (e.g., client code used to lock the client computer) , as taught by Coker et al., to be applicable to any program (e.g., Outlook) on the client computer that would require external server processing. Since a client may send a delete request, for example, to a server that could

take longer than a reasonable time (e.g., server is backed up), it would have been obvious to apply the "lock mechanism" to any program making such a request that would require the client to be locked. During the period in which the client is locked, the user would be informed that a request is taking longer than expected, as taught by Coker et al. Balducci et al. illustrates that a an application program, such as Outlook, may send a request to the server, in turn locking the client during this task, as taught by Coker et al. (e.g., as in the case of emptying a folder on the server or synchronizing a client and server)

However, Coker et al. does not teach a) determining whether at least a threshold period of time has already elapsed while waiting for each of the communications between the client device and the server device to finish for any of the communications. Coker et al. does teach a) providing a message using the code to be presented to a user of the client device [0309-0310] e.g., the code is configured to provide a status indicator) but is silent as to a message displayed in response to the above limitations. Kinoshita et al. teaches the pertinent problem of measuring a time lapse in response to a transmission of a request after a predetermined time, i.e., threshold period, then the displaying a message notifying a user indicating a process is aborted ([0187]) upon failing to receive a result after the predetermined time. Coker et al. teaches sending a request a server, locking the client interface, and informing the client the request may take a long time ([0309])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to set a predetermined amount of time, i.e., threshold period, for the request to be processed by a server, and upon the request failing to processed in the predetermined time, notifying the user of the state of the request. Coker et al. teaches that a user may wait for an

undisclosed period of time while a request is processing in addition to having the user interface locked. Since a predetermined time is indicative of a how long a process should take to complete, it is obvious to notify a user that the process will not complete in the predetermined so as to apprise the user of the status of their request.

17. As per claim 12, Coker et al. teaches the method of claim 11, wherein the presented message is an over-definition of a default message e.g., supra claim 4 discussion)

18. As per claim 13, Coker et al. teaches the method of claim 11, further comprising setting the specific time based on at least one selected from the group consisting of: a roundtrip time for a communication between the server device and the client device, typical roundtrip times for communication between the server device and the client device, a roundtrip time expected by at least one user of the client device, and combinations thereof (supra claim 9)

19. As per claim 14, Coker et al. teaches a computer program product containing executable instructions that when executed cause a processor to perform operations comprising:

block a client device from receiving user input during its communications with a server device ([0309] e.g., locking user interface);

determine whether any of the communications lasts longer than a specific time;

cause a message provided in the computer program product to be presented to a user of the client device if determining that any of the communications lasts longer than the specific time ([0309-0310]);

wherein the computer program product is configured to be provided from the server device to the client device, be stored on the client device and to be executed during each of the communications between the client device and the server device (e.g., supra claim 1)

20. As per claim 15, Coker et al. teaches a computer system comprising:

a server device with server-side framework code which when executed on the server device establishes a client-server framework for client-server communications ([Fig 2], [Fig 13] e.g., client-server communications with executable code),; and

code (i) is configured to be stored on the client device and be executed during each of subsequent communications between the client device and the server device ([0307-0309] e.g., client includes a component called a busy state manager configured to monitor and inform a user of a status and progress of the submitted request), and (ii) when executed blocks the client device from receiving user input during the communications between the client device and the server device ([0309] e.g., client can inform the user that request processing has started and lock the user interface), determines whether any of the communications between the client device and the server device lasts longer than a specific time, and, upon determining that the specific time has been exceeded, causes a message provided in the code to be presented to a user of the client device ([0309-0310] e.g., upon determining that the request from the client may take a long time to process, the server will notify the client accordingly....the client can update the progress bar to show how much of the task has been completed at that point in time.

However, Coker et al. does not teach the server providing executable code to the client computer, i.e., providing the busy-state manager component to a client computer). Lee et al. teaches a system for automatically installing software on a client via a server ([ABSTRACT], [FIG 1])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to install client components using software stored on a server. Lee et al. teaches that

enterprises employ client-server models to facilitate the configuration of a client computer via downloading necessary application software ([0004-0006]). Coker et al. teaches that various types of clients can be supported....the various types of clients including remote clients ([0063]), and in addition, teaches that clients can download a subset of server's data to use locally ([0070]). Therefore, in client-server interactions, as taught by Lee et al. it would have been obvious to enable a server to provide necessary components to remote clients to facilitate server-client interactions, including downloading necessary components to a client. Here, it would have been obvious to download a busy-state manager component to a remote client via an application server.

However, Coker et al., as modified, does not teach that the client device is configured to engage in communications with the server device for a plurality of application programs. Coker et al. does teach that the client computer makes requests to the server such that particular tasks may be executed on the server, and in response to the client request the client computer is locked ([0309-0310] e.g., it is interpreted that the client request is not tied to any particular program such that any program on the client computer that should make a request to the server for server processing would result in the client computer becoming locked). Balducci et al. teaches an application program, i.e., Microslft Outlook, on the client computer ([0024]) and in addition teaches that the client may delete content within the server ([0075]).

Therefore, it would have been obvious to one of ordinary skill in the art to apply the "lock mechanism" (e.g., client code used to lock the client computer) , as taught by Coker et al., to be applicable to any program (e.g., Outlook) on the client computer that would require external server processing. Since a client may send a delete request, for example, to a server that could

take longer than a reasonable time (e.g., server is backed up), it would have been obvious to apply the "lock mechanism" to any program making such a request that would require the client to be locked. During the period in which the client is locked, the user would be informed that a request is taking longer than expected, as taught by Coker et al. Balducci et al. illustrates that a an application program, such as Outlook, may send a request to the server, in turn locking the client during this task, as taught by Coker et al. (e.g., as in the case of emptying a folder on the server or synchronizing a client and server)

However, Coker et al. does not teach determining whether at least a threshold period of time has already elapsed while waiting for each of the communications between the client device and the server device to finish and upon determining that the threshold period of time has already elapsed, causes a message provided in the code to be presented to a user of the client device. Coker et al. does teach a) proving a message using the code to be presented to a user of the client device [0309-0310] e.g., the code is configured to provide a status indicator) but is silent as to a message displayed in response to the above limitations. Kinoshita et al. teaches the pertinent problem of measuring a time lapse in response to a transmission of a request after a predetermined time, i.e., threshold period, then the displaying a message notifying a user indicating a process is aborted ([0187]) upon failing to receive a result after the predetermined time. Coker et al. teaches sending a request a server, locking the client interface, and informing the client the request may take a long time ([0309])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to set a predetermine amount of time for the request to be processed by a server, and upon the request failing to processed in the predetermined time, notifying the user of the state of

the request. Coker et al. teaches that a user may wait for an undisclosed period of time while a request is processing in addition to having the user interface locked. Since a predetermined time is indicative of a how long a process should take to complete, it is obvious to notify a user that the process will not complete in the predetermined so as to apprise the user of the status of their request.

21. As per claim 16, Coker et al. teaches the method of claim 15, wherein a communication lasts longer than the specific time due to network delays, server-side delays, or combinations thereof (supra claim 5)

22. As per claim 17, Coker et al. teaches the method of claim 15, wherein the presented message is an over-definition of a default message (e.g., supra claim 4 discussion)

23. As per claim 18, Coker et al. teaches the computer system of claim 15, wherein the client-side framework code causes the message to be displayed on the client device ([0306-0310] e.g., busy state manager)

24. As per claim 19, Coker et al. teaches the computer system of claim 15, wherein the specific time is based on at least one selected from the group consisting of: typical roundtrip times for communication between the server device and the client device, a roundtrip time expected by at least one user of the client device, and combinations thereof (supra claim 9)

25. As per claim 20, Coker et al., as modified, teaches the computer system of claim 15, wherein at least one roundtrip time for communication between the server device and the client device is recorded (supra Logston et al., Col 25 lines 55-67] e.g., measuring latency or "sent time.") but does not teach that the threshold period of time is set based on the at least one roundtrip time. Kinoshita et al. teaches using a predetermined time used to measure the start of

a request until completion of the request. Conrad teaches setting a time value using the measured latency ([ABSTRACT])

Therefore, at the time the invention was made, one of ordinary skill in the art would have motivation to set a predetermined time from which a request is terminated, as per Kinoshita et al., to take into account measured latencies associated with the request. Since a predetermined time, i.e. ,threshold period of time, determines whether to abort a process, it is obvious to set the predetermined time on latencies to take into these time factors to avoid understating the time allocated for processing.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to DARRIN DUNN whose telephone number is (571)270-1645. The examiner can normally be reached on EST:M-R(8:00-5:00) 9/5/4.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Albert DeCady can be reached on (571) 272-3819. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/DD/
09/19/10

/Albert DeCady/
Supervisory Patent Examiner
Art Unit 2121